Discrete Applied Mathematics 22 (1988/89) 181–189
North-Holland

ACYCLIC k-CONNECTED SUBGRAPHS FOR DISTRIBUTED ALTERNATE ROUTING IN COMMUNICATIONS NETWORKS

Moshe SEGAL
AT&T Bell Laboratories, Holmdel, NJ 07733, USA

Donald M. TOPKIS
Graduate School of Management, University of California, Davis, CA 95616, USA

Received 15 December 1986
Revised 1 October 1987

In a communications network with a history-independent distributed routing control for calls to a particular destination, performance is enhanced if there is a sufficiently diverse alternate routing capability and if cycles are avoided while a path is created. Such a routing relies on the construction of an acyclic spanning subgraph in which all except k nodes are k-connected to the destination node. A linear-time algorithm generates a subgraph with these properties if such a subgraph exists. These subgraphs are characterized, and, whenever the conditions of the characterization are satisfied, a linear-time algorithm constructs k node-disjoint paths from a particular source node to the destination.

1. Introduction

Consider the routing of an arbitrary call to a destination node d in a communications network. The network is represented by a directed graph $G=(N,E)$, with a set of n nodes N and a set of e directed edges E. (To simplify subsequent complexity statements, suppose for convenience that $O(e)\geq n$.) There are no parallel edges (that is, at most one edge exists from any particular node to another). For $x\in N$, $A^G(x)$ is the set of nodes y such that there is a directed edge from x to y in G, and $B^G(x)=\{y: x\in A^G(y)\}$. A *path* from x to y in G is a sequence of nodes $z_0,\ldots,z_m$ such that $z_0=x$, $z_m=y$, and $z_i\in A^G(z_{i-1})$ for $i=1,2,\ldots,m$. The routing of a call involves finding a path in G from the source of the call to d. A *d-routing control* specifies the mechanism for attempting to find such a path. The d-routing control is *distributed* with the path created one node at a time so that, having reached a node $x\neq d$, node x chooses some $y\in A^G(x)$ to append to the path, and the call then proceeds similarly from y. The distributed d-routing control is *history-independent*, so the choice of a node to follow after any given node on a path depends only on the destination d. A *subgraph* of (N,E) is a graph (N',E') with $N'\subseteq N$ and $E'\subseteq E$. A *spanning subgraph* of (N,E) is a graph (N,E') with $E'\subseteq E$. (If S is a spanning subgraph of G, then $A^S(x)\subseteq A^G(x)$ and $B^S(x)\subseteq B^G(x)$ for each $x\in N$.) A particular

history-independent distributed d-routing control can be represented by a spanning subgraph S of G, so that node y may directly follow node x on the path of a call routed to d if and only if $y \in A^S(x)$. Such a subgraph is a *d-routing*. We are interested in constructing a d-routing with certain properties specified below.

A capability for *alternate routing* exists when $A^S(x)$ contains more than one element. A d-routing may enhance survivability, availability, and efficiency if it provides a sufficiently diverse alternate routing capability to mitigate unpredictable damage, failure, and congestion in the network. Two paths are *node-disjoint* if no node, except perhaps the initial and final nodes, appears on both paths. For a positive integer k, a node x is *k-connected to d* in a subgraph S if there exist k paths from x to d in S such that the k paths are pairwise node-disjoint. The *out-degree* of a node x in a subgraph S is $|A^S(x)|$. (For a finite set Z, $|Z|$ is the number of elements in Z.) A node x is *concisely k-connected to d* in S if x has out-degree at least k in S and for *any* $A \subseteq A^S(x)$ with $|A| = k$ there exist k pairwise node-disjoint paths from x to d such that the k nodes directly following x on the k paths are the nodes of A. A set of nodes is *(concisely) k-connected to d* if each node in that set is (concisely) k-connected to d. For a given k, concise k-connectivity from the largest possible set to d is a desirable property to require in a d-routing.

A *cycle* is a path from a node x to itself, where x appears exactly twice on the path and no node other than x appears more than once. A graph is *acyclic* if it contains no cycle. A history-independent distributed d-routing control could cause a path to inefficiently include cycles on its way to d or even to travel around some cycle interminably without ever reaching d. We therefore seek a d-routing that is acyclic. Various methods have been proposed for avoiding cycles in communications networks [1, 5, 10–12]. These include using information (other than the identity of d) transmitted from node to node as the path is generated; having end-to-end control with the entire path chosen at the source; maintaining control of the path from one centralized point; constraining the path to include at most one node between source and destination; using a fixed hierarchy among the nodes to limit path choices; and requiring each node on the path to be closer, in some sense, to the destination than the previous node. The first four methods are inconsistent with the assumption of a history-independent distributed d-routing control. The latter three methods may unduly eliminate possibilities for alternate routing. The present approach has the flavor of the last method, but does not unnecessarily restrict the routing possibilities.

The acyclic property inherently limits k-connectivity. An ordering $x_1, \ldots, x_h$ of the nodes of a subgraph S is an *acyclic ordering* if $A^S(x_i) \subseteq \{x_1, \ldots, x_{i-1}\}$ for $i = 1, \ldots, h$. A subgraph S is acyclic if and only if there exists an acyclic ordering of its nodes [9, p. 29]. Therefore, an acyclic d-routing can have at most $n-k$ nodes with out-degree at least k and hence at most $n-k$ nodes that are k-connected to d, so at least k nodes cannot be k-connected to d. In view of this limitation, assume throughout that we are given as input a predetermined set Q of k nodes such that k-connectivity is not sought from nodes in Q to d. (The set Q is taken here as fixed

and given. Its initial selection remains as a separate issue, and is discussed briefly in Section 4.) The d-routing construction problem now becomes: given $G=(N,E)$, integer k with $1 \leq k \leq n-1$, and $Q \subseteq N$ with $d \in Q$ and $|Q|=k$, find an acyclic spanning subgraph in which $N \setminus Q$ is concisely k-connected to d if such a subgraph exists. (For sets Y and Z, $Y \setminus Z$ is the set of elements in Y but not in Z.)

Section 2 gives an algorithm which constructs in $O(e)$ time a spanning subgraph S and an acyclic ordering $x_1,\ldots,x_n$ of its nodes such that $x_1=d$, $\{x_2,\ldots,x_k\}=Q \setminus d \subseteq B^S(d)$, and each node in $N \setminus Q$ has out-degree at least k in S, if such a subgraph exists. (A single element y considered as a set is denoted y, rather than $\{y\}$.) Section 3 characterizes acyclic spanning subgraphs in which $N \setminus Q$ is concisely k-connected ($k<n$) to d as those spanning subgraphs S for which there is an acyclic ordering of the nodes $x_1,\ldots,x_n$ with $x_1=d$, $\{x_2,\ldots,x_k\}=Q/d \subseteq B^S(d)$, and each node in $N \setminus Q$ has out-degree at least k in S. This result is based on an algorithm which, given an input satisfying the conditions of the characterization, constructs k node-disjoint paths from any node in $N \setminus Q$ to d in $O(e)$ time. A consequence of this characterization is that the $O(e)$ algorithm of Section 2 constructs an acyclic spanning subgraph in which $N \setminus Q$ is concisely k-connected to d, if such a subgraph exists. Section 4 observes that a natural distance-based approach may fail to find a suitable d-routing when one exists, and discusses complexity issues related to the initial section of the given set Q.

Efficient algorithms have been given for various problems involving k-connectivity. These include testing if a graph is k-connected [2, 8]; determining the connectivity of a graph [3]; finding the 2-connected and 3-connected components of a graph [6, 7, 15]; constructing k node-disjoint paths of minimum total cost between two specified nodes [13]; and constructing 2 node-disjoint paths of minimum total cost from one node to each other node [14]. There does not appear to be a previous study of acyclic spanning subgraphs with particular k-connectivity properties.

2. Algorithm

This section gives an efficient algorithm for constructing a spanning subgraph S and an acyclic ordering $x_1,\ldots,x_n$ of its nodes such that $x_1=d$, $\{x_2,\ldots,x_k\}=Q \setminus d \subseteq B^S(d)$, and each node in $N \setminus Q$ has out-degree at least k in S, if a subgraph exists. The input of this algorithm, Algorithm 1, consists of n; the set $B^G(y)$ for each $y \in N$; k; and an ordered set of k nodes $Q=\{x_1,\ldots,x_k\}$ with $x_1=d$. As output the algorithm gives sets $A^S(y)$ for each $y \in N$ (determining an acyclic spanning subgraph S) and an ordered set of h nodes $x_1,\ldots,x_h$ (where $k \leq h \leq n$) such that $Q \setminus d \subseteq B^S(d)$ if $h>k$, $A^S(x_i) \subseteq \{x_1,\ldots,x_{i-1}\}$ for $i=1,\ldots,h$, and x_i has out-degree at least k in S for $i=k+1,\ldots,h$. Theorem 1 shows that this subgraph S has the desired properties (and $h=n$), if any such spanning subgraph exists. The method of the algorithm involves maintaining and iteratively extending an ordering of a subset of the nodes $x_1,\ldots,x_h$. At each iteration, the algorithm picks $y \in N \setminus \{x_1,\ldots,x_h\}$ with

$|A^G(y) \cap \{x_1, \ldots, x_h\}| \geq k$, if such y exists, and sets $x_{h+1} \leftarrow y$, $A^S(x_{h+1}) \leftarrow A^G(x_{h+1}) \cap \{x_1, \ldots, x_h\}$, and $h \leftarrow h+1$. Among the variables used in Algorithm 1, h is the number of nodes already ordered; $A^S(y)$ is the temporary value of that output set for $y \in N \setminus \{x_1, \ldots, x_h\}$; $f(y) = |A^S(y)|$; and R is the set of nodes y with $f(y) \geq k$ and y not yet in the ordering.

Algorithm 1.

Initialization step.

(a) $A^S(x_i) \leftarrow \{x_j\colon 1 \leq j < i$ and $x_i \in B^G(x_j)\}$ for $i = 1, \ldots, k$.

(b) $A^S(y) \leftarrow \{x_i\colon 1 \leq i \leq k$ and $y \in B^G(x_i)\}$ for $y \in N \setminus \{x_1, \ldots, x_k\}$.

(c) $f(y) \leftarrow |A^S(y)|$ for $y \in N \setminus \{x_1, \ldots, x_k\}$.

(d) $R \leftarrow \{y\colon y \in N \setminus \{x_1, \ldots, x_k\}$ and $f(y) = k\}$.

(e) $h \leftarrow k$.

(f) If $\{x_2, \ldots, x_k\} \setminus B^G(d) \neq \emptyset$ then stop.

General step.

(a) If $R = \emptyset$ then stop.

(b) Otherwise

(b.1) select any $z \in R$;

(b.2) $x_{h+1} \leftarrow z$;

(b.3) $h \leftarrow h+1$;

(b.4) $R \leftarrow R \setminus z$;

(b.5) for each $y \in B^G(x_h) \setminus \{x_1, \ldots, x_h\}$,

(b.5.1) $A^S(y) \leftarrow A^S(y) \cup x_h$;

(b.5.2) $f(y) \leftarrow f(y)+1$;

(b.5.3) if $y \notin R$ and $f(y) = k$ then $R \leftarrow R \cup y$.

If $M \subseteq N$, then the subgraph of G *induced by* M, denoted $G(M)$, consists of the set of nodes M and the set of all edges in G joining pairs of nodes in M. Theorem 1 interprets the set $\{x_1, \ldots, x_h\}$ generated by Algorithm 1 either as Q if $(Q \setminus d) \setminus B^G(d) \neq \emptyset$ or otherwise as the greatest set M with $Q \subseteq M \subseteq N$ such that $G(M)$ has an acyclic spanning subgraph T with $Q \setminus d \subseteq B^T(d)$ and with each node in $M \setminus Q$ having out-degree at least k in T.

Theorem 1. *Suppose that there are given inputs consisting of* n; *the set* $B^G(y)$ *for each* $y \in N$; k; *and an ordered set of* k *nodes* $Q = \{x_1, \ldots, x_k\}$ *with* $x_1 = d$. *Algorithm* 1 *terminates in time* $O(e)$. *The spanning subgraph* S *whose edges are determined by the sets* $A^S(y)$ *for each* $y \in N$ *is acyclic,* $Q \setminus d \subseteq B^S(d)$ *if* $h > k$, $A^S(x_i) \subseteq \{x_1, \ldots, x_{i-1}\}$ *for* $i = 1, \ldots, h$, *and* x_i *has out-degree at least* k *in* S *for* $i = k+1, \ldots, h$. *If* $Q \subseteq M \subseteq N$ *and* $G(M)$ *has an acyclic spanning subgraph* T *with* $Q \setminus d \subseteq B^T(d)$ *and with each node in* $M \setminus Q$ *having out-degree at least* k *in* T, *then* $M \subseteq \{x_1, \ldots, x_h\}$. *Therefore, for* $k < n$, G *has an acyclic spanning subgraph* T *such that* $Q \setminus d \subseteq B^T(d)$ *and each node in* $N \setminus Q$ *has out-degree at least* k *in* T *if and only if Algorithm* 1 *terminates with* $h = n$, *and, if so, the subgraph* S *satisfies these conditions.*

Proof. The initialization step takes time $O(e)$. If the general step selects $z \in R$ at a particular iteration, then that iteration of the general step takes time $O(|B^G(z)|)$. No node is selected in more than one iteration of the general step. Therefore, all passes through the general step require total time $O(e)$.

At the beginning and end of each iteration of the general step,

$$A^S(y) = A^G(y) \cap \{x_1, \ldots, x_h\} \quad \text{for each } y \in N \setminus \{x_1, \ldots, x_h\}, \tag{1}$$

$$f(y) = |A^S(y)| = |A^G(y) \cap \{x_1, \ldots, x_h\}| \quad \text{for each } y \in N \setminus \{x_1, \ldots, x_h\}, \tag{2}$$

and

$$R = \{y: y \in N \setminus \{x_1, \ldots, x_h\} \text{ and } f(y) \geq k\}. \tag{3}$$

Suppose that Algorithm 1 creates an ordering of h nodes $x_1, \ldots, x_h$ by termination. By (1),

$$A^S(x_i) = A^G(x_i) \cap \{x_1, \ldots, x_{i-1}\} \quad \text{for } i = 1, \ldots, h$$

and S is acyclic. Consider $h > k$. By (a) and (f) of the initialization step, $d \in A^S(x_i)$ for $i = 2, \ldots, k$. By (2) and (3), $|A^S(x_i)| \geq k$ for $i = k+1, \ldots, h$.

If termination occurs at (f) of the initialization step then $(Q \setminus d) \setminus B^G(d) \neq \emptyset$ and the proof is completed. Suppose that termination occurs at (a) of the general step with $R = \emptyset$. By (2) and (3),

$$|A^G(y) \cap \{x_1, \ldots, x_h\}| < k \quad \text{for each } y \in N \setminus \{x_1, \ldots, x_h\}. \tag{4}$$

Suppose that $Q \subseteq M \subseteq N$ with $M \setminus \{x_1, \ldots, x_h\} \neq \emptyset$ and there exist a spanning subgraph T of $G(M)$ and an acyclic ordering of the nodes $z_1, \ldots, z_{|M|}$ in T with $z_i = x_i$ for $i = 1, \ldots, k$ and $|A^T(z_i)| \geq k$ for $i = k+1, \ldots, |M|$. Pick j to be the smallest i with $z_i \in M \setminus \{x_1, \ldots, x_h\}$. Then

$$\begin{aligned} k &\leq |A^T(z_j)| = |A^T(z_j) \cap \{z_1, \ldots, z_{j-1}\}| \\ &\leq |A^T(z_j) \cap \{x_1, \ldots, x_h\}| \leq |A^G(z_j) \cap \{x_1, \ldots, x_h\}| \end{aligned}$$

which contradicts (4). Therefore no such subgraph T and ordering of the nodes $z_1, \ldots, z_{|M|}$ exist, and the proof is completed. □

The time complexity for Algorithm 1 exhibited in Theorem 1 does not depend on k. Indeed, if $k = 1$ then it is sufficient to find a spanning subtree with d at its root, and that has the same $O(e)$ time complexity as the problem for general k.

Instead of having Algorithm 1 terminate if no feasible solution exists for a given k, one may generalize the present approach to solve for the largest $g \leq k$ such that the problem is feasible with g replacing k. Here, assume that $Q = \{x_1, \ldots, x_k\}$ is given with $|A^G(x_i) \cap \{x_1, \ldots, x_{i-1}\}| = i - 1$ for $i = 2, \ldots, k$ Then the generalized problem is to find the largest $g \leq k$ such that an acyclic spanning subgraph S exists with $\{x_2, \ldots, x_g\} \subseteq B^S(d)$ and with each node in $N \setminus \{x_1, \ldots, x_g\}$ having out-degree at least

g in S. Algorithm 1 requires only minor modifications to solve this problem. The algorithm would maintain $R_g = \{y: y \in N \setminus \{x_1, \ldots, x_h\}, f(y) = g\}$ for $1 \leq g < k$, as well as R. Whenever the generalized algorithm finds $R = \emptyset$ in part (a) of the general step with $h < n$, it would set $k \leftarrow k-1$ and $R \leftarrow R_k$ and continue.

3. Characterization

This section gives, in Theorem 2, a characterization of those acyclic spanning subgraphs in which $N \setminus Q$ is concisely k-connected to d. The proof of Theorem 2 is based on the construction of Algorithm 2 which, given an input consistent with the conditions of the characterization, efficiently constructs k pairwise node-disjoint paths from a particular node in $N \setminus Q$ to d.

The input to Algorithm 2 consists of $n; k$; an ordering of the nodes $x_1, \ldots, x_n$ with $x_1 = d$ and $\{x_1, \ldots, x_k\} = Q$; the set $A^S(x_i) \subseteq A^G(x_i)$ for $i = 1, \ldots, n$; and an integer h with $k+1 \leq h \leq n$. (Note that the sets $A^S(x_i)$ determine a spanning subgraph S of G.) As output the algorithm gives k node-disjoint paths emanating from x_h. Theorem 2 shows that each path is from x_h to d if $\{x_2, \ldots, x_k\} \subseteq B^S(d)$ and the ordering is acyclic in S with x_i having out-degree at least k in S for $i = k+1, \ldots, n$. The method of the algorithm involves maintaining and iteratively extending k node-disjoint paths from x_h. The paths are constructed so that, after the initialization of the algorithm, if x_i is the last node on one path and x_j is some node other than the last node on another path then $i < j$. At each iteration, the algorithm chooses the path whose last node has the largest index, and that path is extended by one more node. Among the variables used in Algorithm 2, $P(i)$ is the index identifying the path containing x_i $(1 < i < h)$ if x_i is on any path and $P(i) = 0$ otherwise; $m(p)$ is the number of nodes on path p $(1 \leq p \leq k)$; $I(p, j)$ is the index of the jth node on path p $(2 \leq j \leq m(p),\ 1 \leq p \leq k)$; and u is the maximum of the indices of the final node $(I(p, m(p)))$ on the various paths.

Algorithm 2.

Initialization step.

(a) $P(i) \leftarrow 0$ for $i = 1, \ldots, h-1$.

(b) If $|A^S(x_h)| \geq k$ then select any distinct $x_{i(1)}, \ldots, x_{i(k)}$ from $A^S(x_h)$; otherwise stop.

(c) For $p = 1, \ldots, k$,

(c.1) $I(p, 2) \leftarrow i(p)$;

(c.2) $m(p) \leftarrow 2$;

(c.3) if $i(p) > 1$ then $P(i(p)) \leftarrow p$.

(d) $u \leftarrow \max\{I(1,2), \ldots, I(k,2)\}$.

General step.

(a) Select any $x_i \in A^S(x_u)$ with $P(i) = 0$ if such x_i exists; otherwise stop.

(a.1) $I(P(u), m(P(u)) + 1) \leftarrow i$;

(a.2) $m(P(u)) \leftarrow m(P(u))+1$;

(a.3) if $i>1$ then $P(i) \leftarrow P(u)$.

(b) If there exists some $v<u$ with $P(v)\neq 0$, then $u \leftarrow \max\{v: v<u, P(v)\neq 0\}$ and continue; otherwise $u \leftarrow 1$ and stop.

Theorem 2. *Suppose that there are given inputs consisting of n; a spanning subgraph S; k; and a set of nodes $Q \subseteq N$ with $d \in Q$ and $|Q| = k < n$. Then S is acyclic with $N \setminus Q$ concisely k-connected to d in S if and only if $Q \setminus d \subseteq B^S(d)$ and there is an acyclic ordering in S of the nodes $x_1, \ldots, x_n$ with $\{x_1, \ldots, x_k\} = Q$, $x_1 = d$, and x_i having out-degree at least k in S for $i = k+1, \ldots, n$. When $Q \setminus d \subseteq B^S(d)$ and such an ordering is given, Algorithm 2 constructs k node-disjoint paths from any x_h $(k+1 \leq h \leq n)$ to d in time $O(e)$.*

Proof. Suppose that S is acyclic with $N \setminus Q$ k-connected to d in S. Because S is acyclic, there exists an acyclic ordering in S of the nodes $x_1, \ldots, x_n$. Then the acyclic ordering together with x_i being k-connected to d for each $x_i \in N \setminus Q$ implies that $\{x_1, \ldots, x_k\} = Q$, $x_1 = d$, and $Q \setminus d \subseteq B^S(d)$. Furthermore, k-connectivity from $N \setminus Q$ to d implies $|A^S(x_i)| \geq k$ for $i = k+1, \ldots, n$.

Now suppose that $Q \setminus d \subseteq B^S(d)$ and there is an acyclic ordering in S of the nodes $x_1, \ldots, x_n$ with $\{x_1, \ldots, x_k\} = Q$, $x_1 = d$, and $|A^S(x_i)| \geq k$ for $i = k+1, \ldots, n$. The acyclic ordering implies that S is acyclic. Pick any h with $k+1 \leq h \leq n$. To complete the proof, it suffices to show that Algorithm 2 constructs k node-disjoint paths from x_h to d in time $O(e)$. Let $I(p, 1) = h$ for $p = 1, \ldots, k$.

We first establish that

$$\max\{I(p, m(p)): p = 1, \ldots, k\} = u < \min\{I(p, j): p = 1, \ldots, k \text{ and } j = 1, \ldots, m(p)-1\} \tag{5}$$

at the beginning and end of each full iteration of the general step. By (d) of the initialization step, (5) holds at the beginning of the first iteration of the general step. Now suppose that (5) holds at the beginning of some arbitrary full iteration of the general step, so it suffices to show that (5) holds at the end of that iteration. But (5) then continues to hold by the induction hypothesis and by (a) and (b) of the general step.

Because $u \geq 2$ at the beginning of each iteration of the general step (except perhaps the first iteration which may begin with $u = 1$ and terminate immediately if $k = 1$) and u is decreased by at least 1 in each full iteration of the general step, Algorithm 2 must terminate. Because $|A^S(x_h)| \geq k$ by hypothesis, Algorithm 2 does not stop at (b) of the initialization step. The algorithm does not stop at (a) of the general step for $u \geq k+1$, because then $|A^S(x_u)| \geq k$ by hypothesis and because (5) implies that there are at most $k-1$ indices $i < u$ with $P(i) \neq 0$. The algorithm does not stop at (a) of the general step for $k \geq u \geq 2$, because then $x_1 = d \in A^S(x_u)$ by hypothesis and $P(1) = 0$. Thus Algorithm 2 must stop at (b) in some iteration of the general step with

$u=1$. Because the algorithm terminates with

$$\max\{I(p,m(p)): p=1,\ldots,k\} = u = 1,$$
$$I(p,m(p)) = 1 \quad \text{for } p=1,\ldots,k$$

at termination.

Consider the k paths constructed such that path p consists of the sequence of nodes $x_{I(p,1)},\ldots,x_{I(p,m(p))}$. Each is a path in S from x_h to d. The paths are node-disjoint because, for $1<i<h$, node x_i is on path p if and only if $P(i)=p$.

The initialization step takes $O(h)$ time. An iteration of the general step for a given u takes $O(|A^S(x_u)|)$ time in part (a). All iterations of the general step total no more than $O(h)$ time in part (b). Thus Algorithm 2 takes time $O(e)$. □

4. Remarks

One approach to avoiding cycling in communications networks with history-independent distributed routing is to restrict the d-routing so that each node on a path is, in some sense, closer to the destination than the previous node [5, 10]. Closeness could be determined by the length of a shortest path, given some weights on the edges, from a node to d. Alternatively, it could be determined by picking a particular spanning tree T (not necessarily a subgraph) rooted at d and having distance measured according to the number of edges in the path in T between a node and d. (The present approach leads to a special case of the latter with T being a chain.) A natural choice of weights is 1 for each edge, resulting in a spanning subtree T of minimum hop paths to d. Such an approach does serve to eliminate cycles, but it imposes routing restrictions that may preclude k-connectivity even in graphs that have a suitable k-connected acyclic spanning subgraph. For example, if $B^G(d)=N\setminus d$ and a minimum hop tree is used, then no node would have more than 1 permissible path to d.

The selection of the $k-1$ nodes in $Q\setminus d$ has been taken throughout as predetermined and given. One selection objective, based on maximizing the number of edges joining pairs of nodes from Q in an acyclic spanning subgraph, would seek nodes $x_1,\ldots,x_k$ with $x_1=d$ and $\{x_1,\ldots,x_{i-1}\}\subseteq A^G(x_i)$ for $i=2,\ldots,k$. The question of whether such a set of nodes exists is NP-complete, because the clique problem [4] can be transformed into it. Another objective, based on maximizing the connectivity from $N\setminus Q$ to Q, would seek $k-1$ nodes $Q\setminus d$ such that $N\setminus Q\subseteq \bigcup_{y\in Q} B^G(y)$. The question of whether such a set of nodes exists is NP-complete, because the minimum cover problem [4] can be transformed into it. The complexity of determining whether there exists $Q\subseteq N$ with $d\in Q$, $|Q|=k$, and such that G has an acyclic spanning subgraph S with $N\setminus Q$ concisely k-connected to d in S remains an open question. Given the set Q, the construction of a suitable spanning subgraph of $G(Q)$ to become the subgraph induced by Q of any feasible spanning subgraph S of G is

trivial. It suffices for that subgraph to be acyclic and include an edge from each node in $Q \setminus d$ to d. A further objective, given Q, would be to seek an acyclic spanning subgraph of $G(Q)$ such that there is connectivity in the subgraph between the maximum possible number $\frac{1}{2}k(k-1)$ of ordered pairs of nodes. The question of whether such a subgraph exists is NP-complete, because the Hamiltonian path problem [4] can be transformed into it. Another further objective, given Q, would be to seek an acyclic spanning subgraph of $G(Q)$ such that the subgraph has the largest possible number of edges. The complexity of this problem remains an open question.

Acknowledgment

The authors are grateful to James M. Kelly whose question stimulated this work.

References

[1] G R. Ash, R.H Cardwell and R.P. Murray, Design and optimization of networks with dynamic routing, Bell System Tech J. 60 (1981) 1787–1820.

[2] S. Even, An algorithm for determining whether the connectivity of a graph is at least k, SIAM J. Comput. 4 (1975) 393–396.

[3] S. Even and R.E. Tarjan, Network flow and testing graph connectivity, SIAM J. Comput. 4 (1975) 507–518.

[4] M R. Garey and D S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness (Freeman, San Francisco, CA, 1979)

[5] T.V. Greene and L.C. Brown, Route control in AUTOVON electronic switching centers, IEEE Trans. Commun Tech. 17 (1969) 442–446.

[6] J Hopcroft and R Tarjan, Efficient algorithms for graph manipulation, Comm. ACM 16 (1973) 372–378.

[7] J E. Hopcroft and R.E Tarjan, Dividing a graph into triconnected components, SIAM J Comput 2 (1973) 135–158.

[8] D.J Kleitman, Methods for investigating connectivity of large graphs, IEEE Trans. Circuit Theory 16 (1969) 232–233

[9] E.L Lawler, Combinatorial Optimization: Networks and Matroids (Holt, Rinehart and Winston, New York, 1976).

[10] T. Nishigaki, C. Ikeda and H Ikegaya, Adaptive loop-free routing technique in store-and-forward communication networks, J Inform. Process 1 (1978) 81–84

[11] R F Rey, ed , Engineering and operations in the Bell system, AT&T Bell Laboratories, Murray Hill, NJ (1983).

[12] M Segal, Traffic engineering of communcations networks with a general class of routing schemes, in· Proceedings Fourth International Teletraffic Congress, London (1964)

[13] J.W. Suurballe, Disjoint paths in a network, Networks 4 (1974) 125–145

[14] J W Suurballe and R.E. Tarjan, A quick method for finding shortest pairs of disjoint paths, Networks 14 (1984) 325–336

[15] R Tarjan, Depth-first search and linear graph algorithms, SIAM J Comput. 1 (1972) 146–160